Documentação Trabalho Prático 3 - ALG 1

Aluno: André Luiz Alves Costa

1. Introdução

O problema computacional abordado neste trabalho consiste em determinar o número mínimo de movimentos necessários para que uma rainha de xadrez capture todos os peões presentes em um tabuleiro personalizado. A rainha pode se mover ortogonalmente e diagonalmente, como no jogo tradicional de xadrez, mas existem células intransponíveis que simulam obstáculos no tabuleiro.

Este trabalho exige a implementação de duas soluções:

- Solução Exata (SE): Garante a resposta correta utilizando algoritmos exatos.
- Solução Aproximada (SA): Prioriza eficiência, mesmo que ocasionalmente forneça respostas subótimas.

A análise comparativa entre as duas abordagens considera qualidade das respostas, tempo de execução e complexidade computacional. Também foi calculada a porcentagem média de desvio dos resultados da solução aproximada em relação à solução exata.

---

2. Modelagem

Representação do Problema

O problema foi modelado com base nos seguintes componentes:

1. Tabuleiro:

- Representado por uma matriz `N x M` de caracteres:
    - `'R'`: Posição inicial da rainha.
    - `'P'`: Posições dos peões.
    - `'.'`: Células vazias.
    - `'-'`: Células intransponíveis.

2. Pontos de Interesse:

- Um vetor armazena as posições relevantes no tabuleiro:
    - O primeiro elemento é a posição inicial da rainha.
    - Os demais elementos são as posições dos peões.

3. Matriz de Distâncias (`dist`):

- Uma matriz `(K+1) x (K+1)` que armazena as distâncias mínimas entre cada par de pontos de interesse, calculadas com uma busca em largura (BFS) adaptada para os movimentos da rainha.

4. Condições de Inacessibilidade:

- Caso algum peão seja inacessível (`dist[0][j]` infinito), a solução retorna imediatamente `-1`.

Essa modelagem é compartilhada pelas duas soluções implementadas.

---

3. Solução

3.1 Solução Exata (SE)

A solução exata utiliza Programação Dinâmica sobre subconjuntos (bitmask) para explorar todas as possíveis sequências de captura dos peões, garantindo a resposta ótima.

1. Definição do Estado:

- `dp[mask][i]`: Custo mínimo para visitar o conjunto de peões representado pelo bitmask `mask` e terminar no ponto `i`.

2. Transições:

- Para cada estado `(mask, i)`, tenta-se adicionar um novo peão `j` ainda não visitado, criando um novo estado `(mask | (1 << j), j)`.

3. Inicialização:

- O estado inicial é `dp[0][0] = 0`, representando a rainha na posição inicial sem capturar nenhum peão.

4. Finalização:

- O resultado final é o menor valor em `dp[fullMask][i]` para qualquer `i`, onde `fullMask` representa o bitmask com todos os peões capturados.

Complexidade:

- Tempo: O(K² * 2^K), onde K é o número de peões.
- Espaço: O(K * 2^K).

3.2 Solução Aproximada (SA)

A solução aproximada utiliza uma heurística gulosa baseada no "vizinho mais próximo", priorizando eficiência.

1. Estado Inicial:
- A rainha começa na posição inicial, com nenhuma célula visitada.
2. Heurística:
- Em cada iteração, escolhe-se o peão mais próximo ainda não capturado, considerando as distâncias na matriz `dist`.
3. Atualização do Estado:
- Após capturar o peão mais próximo, a posição atual da rainha é atualizada para a posição desse peão.
4. Finalização:
- O processo termina quando todos os peões forem capturados ou quando não houver mais peões acessíveis.

Complexidade:

- Tempo: O(K²), pois para cada um dos K passos, verifica-se a distância para todos os outros K peões.
- Espaço: O(K²), devido à matriz de distâncias.

---

4. Análise de Complexidade

Aspecto	**SE (Solução Exata)**	**SA (Solução Aproximada)**
Tempo	O(K² * 2^K)	O(K²)
Espaço	O(K * 2^K)	O(K²)
Garantia de Ótimo	Sim	Não
Eficiência	Ineficiente para K > 16	Muito eficiente para K ≤ 16

---

5. Discussão dos Algoritmos

Resultados Obtidos nos Casos de Teste

Os resultados foram obtidos executando ambos os algoritmos em diferentes instâncias de teste. Os tempos de execução foram medidos em milissegundos (ms).

Instância	Peões	Resultado (SE)	Tempo (SE)	Iterações (SE)	Resultado (SA)	Tempo (SA)	Iterações (SA)
instancia_3p	3	8	0.0035 ms	39	8	0.0003 ms	9
instancia_7p	7	22	0.0942 ms	3143	23	0.0008 ms	49
instancia_10p	10	34	0.8572 ms	51210	36	0.001 ms	100
instancia_12p	12	42	4.1802 ms	294924	43	0.0013 ms	144
instancia_15p	15	43	49.9066 ms	3686415	50	0.0019 ms	225
instancia_16p	16	57	110.996 ms	8388624	68	0.0024 ms	256
instancia_17p	17	53	236.052 ms	18939921	53	0.0022 ms	289
instancia_18p	18	51	520.823 ms	42467346	58	0.0029 ms	324

Análise Comparativa

1. Qualidade das Respostas:

- A solução exata (SE) sempre fornece a resposta ótima.
- A solução aproximada (SA) apresenta uma pequena perda de qualidade em algumas instâncias, especialmente conforme o número de peões aumenta. Por exemplo, na instância com 16 peões, o resultado aproximado foi 68 contra o resultado exato de 57, representando uma diferença significativa.

2. Desvio Médio dos Resultados:

- Calculamos o desvio percentual médio dos resultados da solução aproximada em relação à solução exata.
- O desvio médio nos casos de teste foi de aproximadamente 7.76%, indicando que a solução aproximada tende a fornecer resultados ligeiramente maiores, mas ainda próximos ao ótimo.

3. Tempo de Execução:

- A solução exata tem um crescimento exponencial no tempo de execução à medida que o número de peões aumenta. Na instância com 18 peões, o tempo foi de 520.823 ms, enquanto a solução aproximada levou apenas 0.0029 ms.
- A solução aproximada é extremamente eficiente, com tempos de execução baixos e praticamente constantes para todas as instâncias.

4. Escalabilidade:

- A solução exata torna-se inviável para valores de (K > 16), devido ao crescimento exponencial do número de estados e transições.
- A solução aproximada é escalável e adequada para instâncias maiores, mantendo tempos de execução baixos.

---

6. Conclusão

Neste trabalho, implementamos e analisamos duas abordagens para resolver o problema da rainha e dos peões. A solução exata (SE), embora garantidamente ótima, possui alta complexidade e é limitada a instâncias pequenas. Por outro lado, a solução aproximada (SA) é eficiente e prática, apresentando uma boa relação entre qualidade e desempenho.

Com base nos resultados obtidos, podemos concluir que:

- A solução exata é ideal para instâncias pequenas (K <= 16), onde o tempo de execução permanece aceitável.
- A solução aproximada é preferível para instâncias maiores, sacrificando um pouco da qualidade da resposta em troca de uma eficiência significativamente maior.

---

7. Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.

GeeksforGeeks. *Bitmasking and Dynamic Programming | Travelling Salesman Problem*. Disponível em: https://www.geeksforgeeks.org/dsa/bitmasking-dynamic-programming-set-2-tsp/.